|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

|  |  |
| --- | --- |
| ФАКУЛЬТЕТ | «Фундаментальные науки» |
| КАФЕДРА | «Вычислительная математика и математическая физика» |

**ОТЧЁТ**

***К ЛАБОРТАТОРНОЙ РАБОТЕ***

***НА ТЕМУ:***



|  |  |
| --- | --- |
| ***Многомерная оптимизация*** | |
|  | |
|  | |
|  | |
| Дисциплина: | «Программные технологии разработки систем инженерного анализа» |
|  | |

|  |  |  |
| --- | --- | --- |
| Студент ФН11-11М |  | А.А. Пономарёв |
|  | (Подпись, дата) | (И.О.Фамилия) |
| Преподаватель |  | С.Б. Каримов |
|  | (Подпись, дата) | (И.О.Фамилия) |

*2020 г.*

# ­­­­­­Постановка задачи

Реализовать один из методов многомерной оптимизации (минимизации). В качестве целевых функций взять три функции определённые следующим образом. N точек размещены на единичной сфере . - евклидово расстояние между точками A и B. Первая точка имеет координаты (0,0,1), у второй точки зафиксирована координата .

1. F1(P) = - sum(d(Ai, Aj)^2 for i in 1..N for j in 1..N) Сумма квадратов расстояний между точками, взятая за знаком минус

2. F2(P) = sum(1/d(Ai, Aj) for i in 1..N for j in 1..N if i != j) Сумма обратных расстояний между различными точками

3. F3(P) = - min(d(Ai, Aj) for i in 1..N for j in 1..N if i != j) Минимум среди всех расстояний между различными точками, взятый со знаком минус.

Знак минус в функциях 1 и 3 взят, чтобы переформулировать соответствующую задачу поиска максимума к задаче поиска минимума. Примечание Функция 3 не имеет производных в некоторых точках, так же как, например функция f(x) = min(-x,x) = -x if x > 0 else x. Это доставит сложности для алгоритмов, полагающихся на знание градиента функции. Найти решения для N равного 4, 6, 8, 12, 20.

# Реализация

Программа была реализована с помощью языка программирования Python версии 3.8 в скрипте multi\_dimentional\_optimization.py.

На начальном этапе задаётся стартовая точка (обозначим её 1) и шаги по координатам. Затем замораживаем значения всех координат кроме 1-й, вычисляем значения функции в точках и (где – первая координата точки, а – соответственно значение шага по этой координате) и переходим в точку с наименьшим значением функции. В этой точке замораживаем значения всех координат кроме 2-й, вычисляем значения функции в точках и , переходим в точку с наименьшим значением функции и т. д. для всех координат. В случае, если для какой-нибудь координаты значение в исходной точке меньше, чем значения для обоих направлений шага, то шаг по этой координате уменьшается. Когда шаги по всем координатам  станут меньше соответствующих значений алгоритм завершается, и точка 1 признаётся точкой минимума.

Иллюстрация первого этапа для двух координат:

Diagram, schematic

Description automatically generated

Таким образом, проведя исследующий поиск по всем координатам, мы получим новую точку с наименьшим значением функции в окрестности (обозначим её 2). Теперь можно осуществлять переход ко 2 фазе алгоритма.

На этапе поиска по образцу откладывается точка 3 в направлении от 1 к 2 на том же расстоянии. Её координаты получаются по формуле:{\displaystyle {\overline {x}}\_{3}={\overline {x}}\_{1}+\lambda ({\overline {x}}\_{2}-{\overline {x}}\_{1})}

A picture containing text, clock

Description automatically generated

где – тока с номером , – параметр алгоритма, обычно выбирающийся равным 2. Затем в новой точке 3 проводится исследующий поиск, как на 1 фазе алгоритма, за исключением того, что шаг на этой фазе не уменьшается. Если на этой фазе в результате исследующего поиска удалось получить точку 4, отличную от точки 3, то точку 2 переобозначим на 1, а 4 на 2 и повторим поиск по образцу. В случае если не удаётся найти точку 4, отличную от точки 3, то точку 2 переобозначим на точку 1 и повторим 1-ю фазу алгоритма — исследующий поиск.

Иллюстрация второго этапа для двух координат:

Diagram

Description automatically generated

В скобках отмечены имена точек после переобозначения. На иллюстрации хорошо заметно, как алгоритм корректирует своё направление в зависимости от найденных значений функции.

# Результаты

Были реализованы два метода, согласно условиям задачи: nelder mead и pattern search, описанный в предыдущей главе. Результаты выполнения программы представим на рисунке 1.

Text

Description automatically generated

Рисунок 1 – результат выполнения программы